

Построение Dashboard в python с помощью библиотеки Dash

Малик Александр Игоревич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Целью данной статьи является, установка одного из веб-фреймворков в связке с Python под названием Dash и его использование для визуализации данных и проведения последующих анализов. Dash от Plotly — это веб-фреймворк, построенный на основе Plotly.js, React.js и Flask, который можно использовать для создания аналитических веб-приложений. Результатом исследования станет готовый веб-сервер с визуализированной аналитической базой.

Ключевые слова: Dash, PyCharm, Python, Dashboards, Визуализация данных

Building the Dashboard in python using a Dash framework

Malik Aleksandr Igorevich

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of this article is to install one of the most convenient web frameworks in conjunction with Python called Dash and use it for data visualization and subsequent analysis. Dash from Plotly is a web framework built on the basis of Plotly.js, React and Flask, which can be used to create analytical web applications. The result of the study will be a ready-made web server with a visualized analytical database.

Keywords: Dash, PyCharm, Python, Dashboards, Data visualization

1 Введение

1.1 Актуальность

С каждым днем наиболее популярной темой в мире IT становится разработка интерактивных веб-приложений.

Такие веб-приложения нужны для моментальной визуализации данных, также они отлично подходят для работы при анализе данных, что позволяет наглядно, моментально продемонстрировать полученный результат, после ввода и некоторой первичной обработки данных.

1.2 Обзор исследований

В своей работе В.А. Мясоедова рассмотрела два наиболее популярных веб-фреймворка скриптовых языков программирования для создания

интерактивных веб-приложений Dash и Shiny и провела сравнение между ними по определенным критериям [1]. Д.Н. Петриев продемонстрировал возможности библиотеки Dash создав интерактивные расчетные веб-приложения [2]. А. И. Тихонов проанализировал возможность использования технологии Plotly Dash для создания интерактивных веб-приложений, интегрируемых в электронные учебники [3]. Ю. А. Цветков М. А. Котерев - спроектировали и реализовали автоматизацию процесса обработки и анализа данных, полученных с приборов, разрабатываемых на кафедре ПМИВТ, фиксирующих климатические показания и передающие данные измерений на сервер [4]. В. В. Костерин привел результаты поиска инструментальных средств для визуализации результатов исследований и для моделирования социально-экономических процессов, показывающих, что два фреймворка языка программирования Python Bokeh и Dash не требуют углублённой подготовки в программировании и наиболее эффективно решают подобные задачи, позволяя делать интерактивные графические презентации в сети интернет [5]. В данном посте Е. Туренко описывает веб-фреймворк Dash, объясняя работу с ним на примере создания нескольких аналитических дашбордов [6]. М. Маргарита рассказывает о том, как определенные веб-элементы, доступные в приложении Dash, можно использовать для создания интерактивной, детализированной и информативной панели с динамическими параметрами и табличными разделами [7].

1.3 Цель исследования

Цель исследования - установить модуль Dash и использовать его для визуализации и анализа данных на веб-сервере.

2 Материалы и методы

Для исполнения программного кода и экранирования результата на веб-сервер в Python подключается и используется модуль Dash. Используется среда разработки PyCharm Community Edition. Работа будет происходить на готовых данных об авокадо, скачать которые можно по ссылке: <https://drive.google.com/file/d/1hdes6nfGpLpD-satZlWLOyVP9XT0r9Zc/view?usp=sharing>

3 Результаты и обсуждения

Перед началом работы требуется установить библиотеку Dash. **File > Settings... > Project: project.py > Python interpreter**, после чего, нажатием на «+» активируем поисковую строку и вводим **Dash**, далее необходимо выполнить установку нажатием на **Install Package** (рис.1).

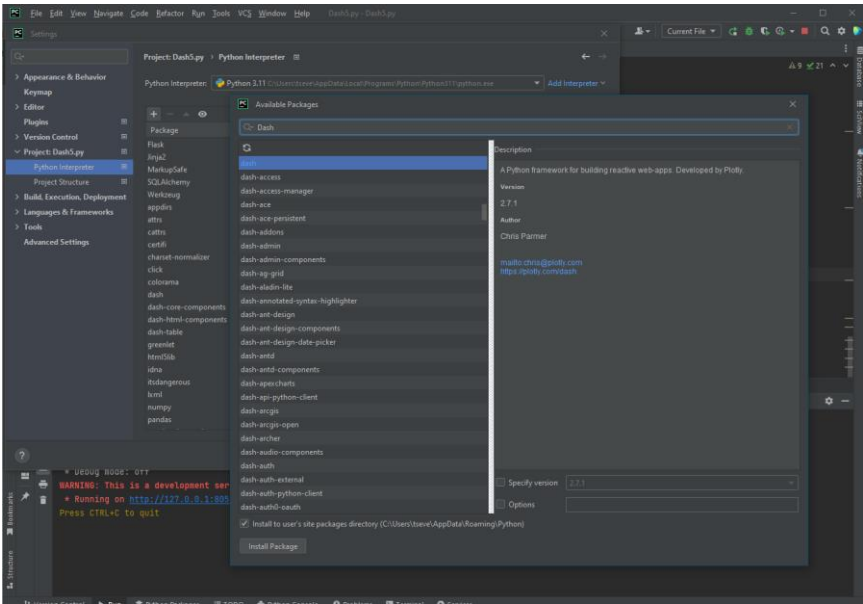


Рис. 1 – Установка библиотеки Dash

Также для работы необходимы 2 дополнительные библиотеки **Pandas** и **Numpy**, которые можно установить таким же способом. Перемещаем файл с данными в папку с исходным кодом и создаем файл `assets` в котором, будет содержаться стиль веб-страницы(`style.css`) (рис.2)

Practice					
	Имя	Дата изменения	Тип	Размер	
	assets	18.12.2022 21:53	Папка с файлами		
	avocado.csv	01.10.2019 9:54	Файл Microsoft E...	1 943 КБ	
	avocado.py	18.12.2022 21:46	JetBrains PyCharm	6 КБ	

Рис. 2 – Директория с работой

Импортируем все необходимые библиотеки (рис.3)

```
import dash
from dash import dcc
from dash import html
import pandas as pd
import numpy as np
from dash.dependencies import Output, Input
```

Рис. 3 – Импорт необходимых библиотек

Вначале прочитаем файл с данными, записав их во фрейм. Объявим переменную **data**, в которую методом датафрейма **read_csv** считаем данные из файла **avocado.csv**, добавим колонку с датой и отсортируем значения по дате (рис.4).

```
data = pd.read_csv("avocado.csv")
data["Date"] = pd.to_datetime(data["Date"], format="%Y-%m-%d")
data.sort_values("Date", inplace=True)
```

Рис. 4 – Получение данных

Далее необходимо подключить стиль **style.css**. Пропишем в него следующий код, определяющий стиль веб-страницы (рис.5).

```
body { font-family: "Lato", sans-serif; margin: 0; background-color: #F7F7F7; }

.header {background-color: #222222; height: 288px; padding: 16px 0 0 0;}

.header-emoji {font-size: 48px; margin: 0 auto; text-align: center;}

.header-title {color: #FFFFFF; font-size: 48px; font-weight: bold; text-align: center; margin: 0 auto;}

.header-description {color: #CFCFCF; margin: 4px auto; text-align: center; max-width: 384px;}

.wrapper {margin-right: auto; margin-left: auto; max-width: 1024px; padding-right: 10px; padding-left: 10px; margin-top: 32px;}

.card {margin-bottom: 24px; box-shadow: 0 4px 6px 0 rgba(0, 0, 0, 0.18);}

.menu {height: 112px; width: 912px; display: flex; justify-content: space-between; padding-top: 24px; margin: -80px auto 0 auto; background-color: #FFFFFF; box-shadow: 0 4px 6px 0 rgba(0, 0, 0, 0.18);}

.Select-control {width: 256px; height: 48px;}

.Select--single > .Select-control .Select-value, .Select-placeholder {line-height: 48px;}

.Select--multi .Select-value-label {line-height: 32px;}

.menu-title {margin-bottom: 6px; font-weight: bold; color: #879A82;}
```

Рис. 5 – Заполнения файла со стилями, нужным стилем

Внедряем стиль непосредственно в будущее веб-приложение, создаем переменную для хранения дашборда и присваиваем наименование странице (рис.6).

```
external_stylesheets = [
    {
        "href": "https://fonts.googleapis.com/css2?"
        "family=Lato:wght@400;700&display=swap",
        "rel": "stylesheet",
    },
]

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
app.title = "Avocado аналитика: Анализируем все!"
```

Рис. 6 – Подключение стилей, создание дашборда

Определяем структуру **html**-страницы при помощи подраздела библиотеки **dash dash_component_html**, для этого пропишем следующий код (рис.7).

```
app.layout = html.Div(  
  children=[  
    html.Div(  
      children=[  
        html.P(children="🥑", className="header-emoji"),  
        html.H1(  
          children="Avocado Analytics", className="header-title"  
        ),  
        html.P(  
          children="Проанализировано поведение цен на авокадо"  
            " и количество авокадо, проданных в США"  
            " между 2015 и 2018 годами",  
          className="header-description",  
        ),  
      ],  
      className="header",  
    ),  
  ],  
)
```

Рис. 7 – Добавление заголовка(header) html-странице

После чего необходимо добавить три блока с визуализацией и фильтрами данных (рис.8).

```

html.Div(
    children=[
        html.Div(
            children=[
                html.Div(children="Region", className="menu-title"),
                dcc.Dropdown(
                    id="region-filter",
                    options=[{"label": region, "value": region} for region in np.sort(data.region.unique())],
                    value="Albany", clearable=False, className="dropdown",
                ),
            ]
        ),
        html.Div(
            children=[
                html.Div(children="Type", className="menu-title"),
                dcc.Dropdown(
                    id="type-filter",
                    options=[{"label": avocado_type, "value": avocado_type} for avocado_type in data.type.unique()],
                    value="organic", clearable=False, searchable=False, className="dropdown",
                ),
            ],
        ),
        html.Div(
            children=[
                html.Div(
                    children="Date Range",
                    className="menu-title"
                ),
                dcc.DatePickerRange(
                    id="date-range",
                    min_date_allowed=data.Date.min().date(),
                    max_date_allowed=data.Date.max().date(),
                    start_date=data.Date.min().date(),
                    end_date=data.Date.max().date(),
                ),
            ],
        ),
    ],
    className="menu",
),

```

Рис. 8 – Добавление 3 блоков: фильтр, дашборд цены, дашборд продаж

Там же прописываем тип отображения дашбордов(рис.9).

```

html.Div(
    children=[
        html.Div(
            children=dcc.Graph(
                id="price-chart", config={"displayModeBar": False},
            ),
            className="card",
        ),
        html.Div(
            children=dcc.Graph(
                id="volume-chart", config={"displayModeBar": False},
            ),
            className="card",
        ),
    ],
    className="wrapper",
),
]
)

```

Рис. 9 – Определение типа отображения дашбордов

Работа над внешней частью подошла к концу, теперь нужно настроить все это дело изнутри: создадим обратный вызов(callback) дашборда (рис.10) и функцию обновления данных в реальном времени (рис.11).

```

@app.callback(
    [Output("price-chart", "figure"), Output("volume-chart", "figure")],
    [
        Input("region-filter", "value"),
        Input("type-filter", "value"),
        Input("date-range", "start_date"),
        Input("date-range", "end_date"),
    ],
)

```

Рис. 10 – Обратный вызов дашборда для обработки и декорирования данных

```
def update_charts(region, avocado_type, start_date, end_date):
    mask = ((data.region == region) & (data.type == avocado_type) & (data.Date >= start_date) & (data.Date <= end_date))
    filtered_data = data.loc[mask, :]
    price_chart_figure = {
        "data": [
            {
                "x": filtered_data["Date"],
                "y": filtered_data["AveragePrice"],
                "type": "lines",
                "hovertemplate": "%{y:.2f}<extra></extra>",
            },
        ],
        "layout": {
            "title": {
                "text": "Average Price of Avocados",
                "x": 0.05,
                "xanchor": "left",
            },
            "xaxis": {"fixedrange": True},
            "yaxis": {"tickprefix": "$", "fixedrange": True},
            "colorway": ["#17B897"],
        },
    }

    volume_chart_figure = {
        "data": [
            {
                "x": filtered_data["Date"],
                "y": filtered_data["Total Volume"],
                "type": "lines",
            },
        ],
        "layout": {
            "title": {"text": "Avocados Sold", "x": 0.05, "xanchor": "left"},
            "xaxis": {"fixedrange": True},
            "yaxis": {"fixedrange": True},
            "colorway": ["#E12039"],
        },
    }

    return price_chart_figure, volume_chart_figure
```

Рис. 11 – Функция обновления данных

После чего остается проверить работоспособность результата запустив виртуальный сервер (рис. 12)

```
if __name__ == "__main__":
    app.run_server(debug=True)
```

Рис. 12 – Развертывание виртуального веб-сервера внутри среды

Если все манипуляции произведены правильно, то в результате компиляции в окне отладки можно увидеть информацию о запущенном сервере, а также ссылку на сам ресурс для просмотра результата работы (рис. 13).

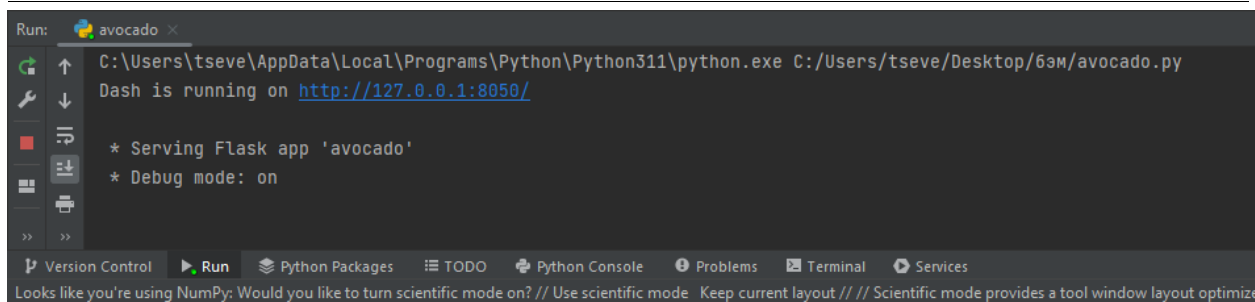


Рис. 13 – Дебаг панель после компиляции

По итогу получается полнофункциональное интерактивное приложение с приведенным анализом данных, в которое в любом момент можно внести нужную правку или изменить данные (рис.14).

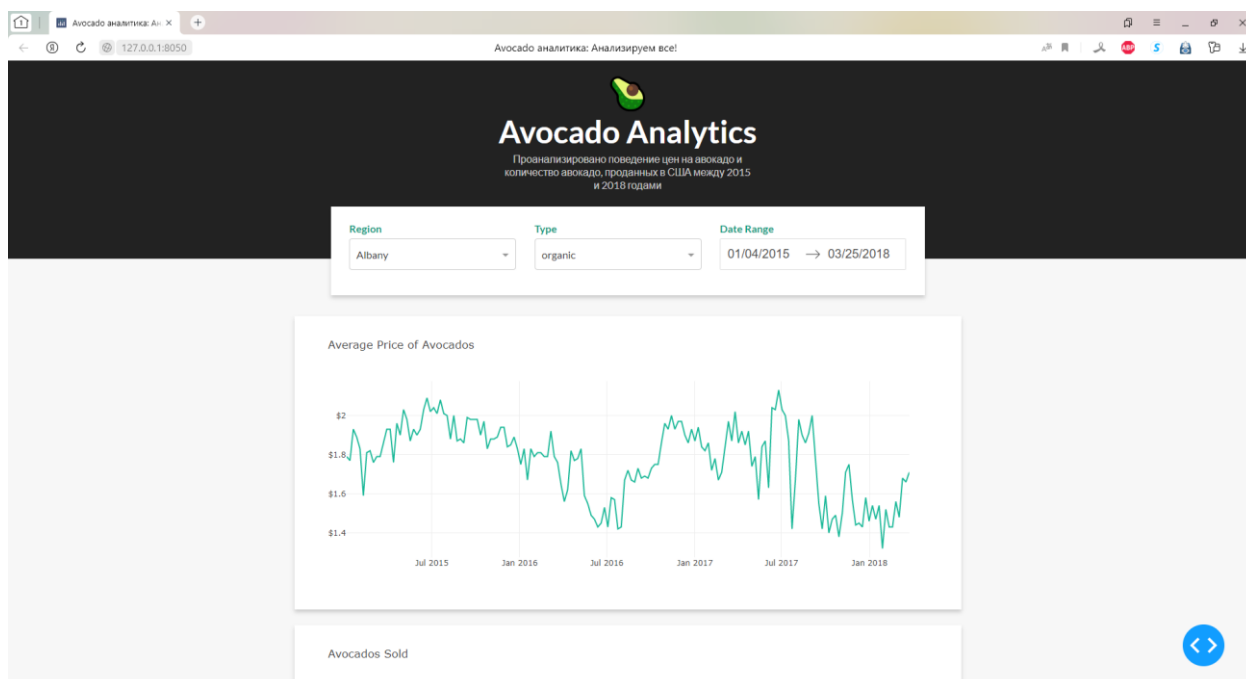


Рис. 14 – Готовое веб приложение при помощи фреймворка Dash

4 Выводы

В данной работе показано простейшей создание визуализации и анализом данных через веб-фреймворк Dash, в которой было рассмотрено взаимодействие библиотеки с внешним html-содержимым, созданием дизайна и обработки внешней информации в любом удобном виде, в результате чего получается готовый веб-продукт для последующей его реализации.

Библиографический список

1. Мясоедова В. А. Сравнение технологий создания интерактивных веб-приложений на примере dash и Shiny //Modern Science. 2021. №. 1-1. С. 479-482.
2. Петриев Д. Н. Интерактивные расчетные веб-приложения на фреймворке

- Dash // Радиоэлектроника, электротехника и энергетика : Тезисы докладов, Москва, 14–15 марта 2019 года. М.: Общество с ограниченной ответственностью "Центр полиграфических услуг " РАДУГА", 2019. С. 332.
3. Тихонов А. И. Интеграция интерактивных веб-приложений в электронные учебники // Современные Web-технологии в цифровом образовании: значение, возможности, реализация : Сборник статей участников V-ой Международной научно-практической конференции, Арзамас, 17–18 мая 2019 года / Научный редактор С.В. Миронова, ответственный редактор С.В. Напалков; Арзамасский филиал федерального государственного автономного образовательного учреждения высшего образования "Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского". Арзамас: Арзамасский филиал федерального государственного автономного образовательного учреждения высшего образования "Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского", 2019. 618 с.
 4. Цветков Ю. А., Котерев М. А. Разработка web-приложения для мониторинга состояния серверов и приборов сети наблюдения МФ // Студенческая научная весна: Всероссийская студенческая конференция: сборник тезисов докладов, Москва, 01–30 апреля 2021 года. М.: Издательский дом "Научная библиотека", 2021. С. 448-449.
 5. Костерин В. В. Интерактивная графика в экономических расчетах на Python // Наука ЮУрГУ: Секции экономики, управления и права : материалы 73-й научной конференции, Челябинск, 20–22 апреля 2021 года. Челябинск: Издательский центр ЮУрГУ. 2021. С. 6-12.
 6. Туренко Е. Создание реактивных аналитических веб-приложений с использованием Python и библиотеки Dash // Создание web-приложений на Python. Питон в веб-разработке. URL: <https://tproger.ru/translations/reactive-web-apps-with-python/> (дата обращения: 18.12.2022)
 7. Margarita M. Работа с панелью индикаторов. Руководство программиста Python. Часть 1. URL: <https://medium.com/nuances-of-programming/работа-с-панелью-индикаторов-руководство-программиста-python-часть-1-cf621282daa7> (дата обращения: 18.12.2022)